NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

DEVELOPING INTEGRATED DECISION SUPPORT SYSTEMS FROM MATHEMATICAL MODELS

by

Michael S. Downs

September, 1995

Principal Advisor:	Hemant Bhargava
Associate Advisor:	R. Kevin Wood

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DEVELOPING INTEGRATED DECISION SUPPORT SYSTEMS FROM MATHEMATICAL MODELS	5. FUNDING NUMBERS
6. AUTHOR(S) Downs, Michael S.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

A methodology for the creation of decision support systems (DSS) from mathematical programming models is examined. This approach is demonstrated using a model for flight scheduling, integrating a formal data model, represented in a database management system (Paradox), and a mathematical programming model, represented in an executable mathematical modeling language (GAMS).

The integration is completed by creating a database capable of supporting the GAMS model, creating a series of queries to extract the data required by the GAMS model, and finally by modifying the GAMS model to import the external data sets and write the results to a file that can be interpreted by the database. This process is first completed manually to support a specific GAMS model. In the second portion of this thesis, the process is generalized to provide a framework that can be used to design a front-end database for any GAMS model.

Benefits of this integrated approach over using a stand alone mathematical model include: The assurance of model integrity, explicit data modeling, improved representation and manipulation of model inputs and outputs, greater integrity of input data, and easier interpretation and multiple views of model outputs.

14. SUBJECT TERMS Decision Support System, Mathematical Models, Relational Database Management Systems, Modeling Languages, Model Manegement	15. NUMBER OF PAGES 85
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

DEVELOPING INTEGRATED DECISION SUPPORT SYSTEMS FROM MATHEMATICAL MODELS

Michael S. Downs
Lieutenant, United States Navy
B.S., University of Missouri, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1995

Author: Michael S. Downs

Approved by: Hemant Bhargava, Principal Advisor

R. Kevin Wood, Associate Advisor

Reuben Harris, Chairman
Department of Systems Management

ABSTRACT

A methodology for the creation of decision support systems (DSS) from mathematical programming models is examined. This approach is demonstrated using a model for flight scheduling, integrating a formal data model, represented in a database management system (Paradox), and a mathematical programming model, represented in an executable mathematical modeling language (GAMS).

The integration is completed by creating a database capable of supporting the GAMS model, creating a series of queries to extract the data required by the GAMS model, and finally by modifying the GAMS model to import the external data sets and write the results to a file that can be interpreted by the database. This process is first completed manually to support a specific GAMS model. In the second portion of this thesis, the process is generalized to provide a framework that can be used to design a front-end database for any GAMS model.

Benefits of this integrated approach over using a stand alone mathematical model include: The assurance of model integrity, explicit data modeling, improved representation and manipulation of model inputs and outputs, greater integrity of input data, and easier interpretation and multiple views of model outputs.

TABLE OF CONTENTS

I. INTRODUCTION 1

II. THE FLIGHT SCHEDULE PROCESS 5

A. INTRODUCTION 5

B. PROBLEM DESCRIPTION 5

1. Inputs to the flight schedule 6

a. Events in the flight syllabus 6

b. Students available for training 6

c. Instructors available for training 7

2. Constraints on the Flight Schedule 7

a. Regulations 7

b. Maximum number of days a student has to complete the syllabus 8

c. Events required to be completed before an event is scheduled. . . 8

d. Flight hours available for the day 8

e. Number of aircraft available for the day 8

f. Sunrise, sunset and other constraints 9

3. Problem description summary 9

C. STEPS IN THE FLIGHT SCHEDULE PROCESS 9

1. Determine mission requirements for the day 9

2. Flight hour availability 10

3. Number of aircraft available 10

4. Assigning missions to aircraft 10

5. Assignment of instructors to events 11

D. CURRENT TECHNOLOGY 11

E. ALTERNATIVE METHODS 11

F. OPTIMIZATION APPROACH TO THE FLIGHT SCHEDULE PROCESS . 12

1. Model integrity 12

2. Data integrity 12

3. Input of data 13

4. Management of multiple input and output sets 13

5. Friendly presentation of data 13

III. DECISION SUPPORT SYSTEM APPROACH 15

A. A COMPARISON OF THE GAMS AND DSS APPROACH 15

1. The GAMS approach 15

2. The DSS approach 17

3. Advantages of the DSS over the GAMS approach 18

B. DESIGN OF THE DATABASE 18

1. An overview of relational database concepts 19

2. Selection of relations from GAMS sets 19

3. Selection of attributes from GAMS parameters 19

4. Selection of relations from GAMS tables 19

5. USMC Trainee Model database design 20

C. LINKS FROM THE DATABASE TO THE MODEL 21

1. Determine the data requirements of the model 21

2. Determine required database queries 21

3. Conversion of database data to mathematical model inputs. 22

D. LINKS FROM THE MODELBASE TO THE DATABASE 22

E. DESIGN OF THE DIALOGUE GENERATOR 23

1. Data entry 24

2. Data modification 25

3. Data deletion 25

4. Bounds checking 25

F. ADVANTAGES 26

1. Model integrity 26

2. Input of data 26

3. Data integrity 27

4. Automated modification of dynamic data 27

5. Alternative views of data 28

6. Management of multiple input and output data sets 28

7. Friendly presentation of solutions 28

G. SUMMARY 29

IV. AUTOMATION OF THE DSS APPROACH 31

A. CAPTURE OF THE GAMS MODEL METADATA 31

B. DATABASE DESIGN 32

C. CREATION OF QUERIES 33

D. LINKS FROM THE DATABASE TO THE MODELBASE 35

E. LINKS FROM THE MODELBASE TO THE DATABASE 36

F. SUMMARY 37

V. CONCLUSION AND RECOMMENDATIONS 39

LIST OF REFERENCES 41

BIBLIOGRAPHY 43

APPENDIX A. MATHEMATICAL DESCRIPTION OF USMC TRAINEE MODEL . . 45

APPENDIX B. USMC TRAINEE MODEL DATA SETS AND OUTPUT ROUTINES 51

APPENDIX C. GAMS DATA ELEMENTS FOR THE USMC TRAINEE MODEL . . 59

APPENDIX D. USMC TRAINEE MODEL IN THIRD NORMAL FORM 61

APPENDIX E. USMC TRAINEE GAMS MODEL REQUIRED INPUTS 63

APPENDIX F. REQUIRED QUERIES 65

APPENDIX G. MODIFIED USMC GAMS TRAINEE MODEL DATA SETS AND OUTPUT ROUTINES 67

APPENDIX H. HIERARCHICAL MENU STRUCTURE 71

INITIAL DISTRIBUTION LIST 73

I. INTRODUCTION

Decision support systems (DSS) support decision making by facilitating the use of data and models. These systems typically have three components: a database, a model base, and a user interface[Ref. 1]. The critical and distinctive feature of a DSS is the use of mathematical or other models in the decision-making process. Few available DSS generators[Ref. 1] have sufficient features for model representation, execution and management. Serious modelers therefore use a variety of model-oriented approaches such as executable algebraic modeling languages. One representative and popular language in this category is GAMS (General Algebraic Modeling Language)[Ref. 2]. GAMS not only provides an algebraic language for representing mathematical models, it also provides automatic translators to a number of solvers for optimization models.

The purpose of this thesis is to: (1) Integrate a GAMS model and a Paradox relational database to create a prototype DSS, and (2) generalize this approach into a set of rules that can be used to produce a computer program capable of generating a set of database relations (tables), and links required to integrate a GAMS model into a DSS. GAMS and Paradox were selected for this thesis because they are widely used by students at the Naval Postgraduate School.

The GAMS model used in this thesis is the USMC Trainee model written by Kawakami[Ref. 3]. It is a prototype mathematical optimization model written to support the flight scheduling process at a United States Marine Corps (USMC) fleet replacement squadron (FRS). This model was selected because it appears to provide good solutions to the FRS flight scheduling problem. A second reason for selecting this model is that it is a real model developed for the flight scheduling process that represents the level of complexity that a flight scheduling DSS must be able to support. The DSS presented in this thesis is a prototype to show a method of linking GAMS models into a DSS. If this specific DSS were implemented, it would be used by the flight scheduling officer at a FRS to assist in writing the daily flight schedule.

In addition, the DSS generation techniques presented are appropriate for linking any GAMS model into a DSS.

In model-oriented languages such as GAMS, the "model declaration" consists of a rudimentary "data definition," mathematical model, and control statements for presentation of the output. Few guidelines or design principles -- well known and accepted in the database management literature -- for data definition are enforced in such languages. This lack of a formal data model and management system for the data schema and the database, leads to several problems, including those concerned with model maintenance, data integrity, alternative views of the input data, management of multiple input and output data sets, and meaningful presentation of solutions.

Model integrity within a GAMS model cannot be guaranteed because the data and mathematical model are contained in a single GAMS file. A user attempting to modify a model's data may accidentally modify the mathematical model. Data integrity and the input of data are both problematic because data stored in a GAMS file are polyinstantiated, i.e., attributes of an instance of an element are held in more than one table. Alternative views of data is not possible because GAMS requires that input data be stored in a singularly prescribed format. Management of multiple input and output data sets is difficult and resource intensive because the data and model are stored together. GAMS output capabilities are limited and can be problematic for a user not familiar with the GAMS language.

There has been a great deal of research on the use of executable modeling languages as the model base component of a DSS. Bhargava, Krishnan and Mukherjee[Ref. 4] examine ways that data modeling features can enhance the capabilities of mathematical modeling languages. They note that developers of mathematical models and modeling systems largely ignore data modeling features. This presents several problems for the user of mathematical models. One problem is that the developers of mathematical models view data differently than the users. Krishnan noted that "potential users of mathematical modeling techniques find it

easier to conceptualize a problem in terms of the data modeling relationships rather than mathematical relationships."[Ref. 5] Another problem is that mathematical models that store data with the model have no way of accessing external data sources. It is usually preferable to access data for a mathematical model from an existing, up-to-date database rather than creating and maintaining a set of data solely for the model being used[Ref. 5].

To illustrate the process of integrating a mathematical model into a DSS, the USMC Trainee Model will be integrated with a Paradox relational database. This integration is completed by developing a database to support the GAMS model, replacing the GAMS data with calls to external files created by the database, and exporting the results of the GAMS model to the database. The control mechanisms for both the database and GAMS model are encapsulated in a user interface that controls the creation of data files, execution of the GAMS model and extraction of the solution from the solver's report. This approach alleviates the problems referred to above and results in a complete, integrated DSS.

In Chapter II, the flight schedule process and proposed solutions to the flight scheduling problem are presented. In Chapter III, an integrated DSS solution using the USMC Trainee Model[Ref. 3] is presented. The DSS solution presented in Chapter III is achieved with a tremendous amount of programming which would not be particularly useful if it were not generalizable. Chapter IV presents a systematic, generalized version of the DSS approach. While not fully implemented due to time constraints, in theory this approach can be used to create a computer automated tool to create the database and all linkages required to integrate a GAMS model into a DSS. This approach shows that the programs required to develop the integrated system can themselves be generated using a set of higher-level programs. In theory, this can be accomplished requiring only the data definition portion of the "model" declaration.

Kawakami noted that to be used in practice, the USMC Trainee model would require the implementation of an efficient database management system[Ref. 3].

Integrating Kawakami's model into a DSS fulfills this requirement and brings the model one step closer to being ready to use by an aviation squadron's flight scheduler. The significance of developing an automated tool that generates an integrated database-GAMS DSS is that it will allow GAMS models to be more accessible to users who are not necessarily interested in the workings of the mathematical model, but rather in the inputs to the model and the model's results.

II. THE FLIGHT SCHEDULE PROCESS

This chapter describes the flight scheduling process, inputs to and constraints on the flight schedule, and approaches taken to automate the flight scheduling process. This background is necessary to provide an understanding of the complexity of the task faced by the flight scheduler on a daily basis in writing the flight schedule.

A. INTRODUCTION

In a military aviation squadron, the flight schedule is used as the central planning document around which all daily activities are planned. It lists missions to be conducted, the pilots who will fly the missions, and aircraft that will be used. For an FRS, the inputs to the flight scheduling process are the training events to be accomplished, students available to fly, and instructors available to teach. The output of the flight schedule process is the flight schedule, listing every event that the squadron will fly on a given day. Every event listed on the flight schedule has an event number, launch time, recovery time, aircrew, aircraft to be flown, and other notes concerning the flight.

B. PROBLEM DESCRIPTION

In writing a flight schedule, the scheduler attempts to create a schedule that meets all of the squadron's assigned missions constrained by the number of pilots, aircraft and flight hours available for a given day. The number of different flight schedules that can be written for even small sets of pilots can grow large quickly.

As an example, assume a training squadron (FRS) has 15 instructors and 15 students and that each instructor and student will fly exactly once on a given day. The scheduler can fly the first student, "student one" with any of the 15 instructors. Once "student one" has been assigned, "student two" can then be paired with any of the 14 remaining instructors, yielding 15 X 14 = 210 possible instructor-student pairings for the first two students. Continuing this construction shows that the scheduler has $15! = 1.3 \times 10^{12}$ different pairings of students and instructors that

he/she can use on the flight schedule. If the scheduler is constrained to flying say 12 flights, he/she must select a set of 12 crews from this large set of possible crews, a task that is humanly infeasible on a daily basis. A second problem that the scheduler faces when considering which crews to fly is that he/she has no objective way of deciding which sets of crews will make the best use of the aircraft, flight hours and crews for a given day.

Given the large number of alternatives that the scheduler must consider and the fact that there is no objective criteria for determining the efficiency of a given schedule, the schedule is normally completed using satisficing rules, (settling for a less-than-optimal solution)[Ref. 8] rather than optimization rules. Mathematical models provide the scheduler a tool to provide objective measures of schedule efficiency while simplifying and partially automating the daily scheduling process. To illustrate the true complexity of the flight scheduling process, the inputs, constraints and process used to create a flight schedule will be discussed.

1. Inputs to the flight schedule

For an FRS, the squadron's mission is to train aviators to fly a specific model of aircraft. The inputs to the flight scheduling process are the training events to be accomplished, students available to fly, and instructors available to teach.

a. Events in the flight syllabus

The specific requirements that a student must complete prior to being certified in a specific model of aircraft, i.e., H-53, H-60B, are defined in the FRS syllabus. This certification requires that a pilot complete all prescribed syllabus flights (events). Each event has attributes: The event name, event duration and required prerequisite events.

b. Students available for training

On a given day, not all students will be available for training. A student may be on leave, have ground training scheduled for the day, or be medically excluded from flying. Students have attributes that the scheduler must be aware of.

These attributes are: The student's name, events that the student has flown, and the number of days a student has been in the syllabus.

c. Instructors available for training

For the same reasons that all students may not be available to fly, instructors are not always available to instruct. The attributes associated with an instructor are: The instructor's name, events that the instructor is qualified to teach, and the number of hours that the instructor is available to fly.

2. Constraints on the flight schedule

The flight schedule is constrained by regulations and the scarcity of resources. Regulations concerning the conduct of training come from many sources and exist to ensure that the training is conducted safely. Other constraints are imposed by the limitation of resources, such as the number of flight hours allocated to the squadron, or the length of time that a student is assigned to the FRS.

a. Regulations

There are many regulations that the scheduler must consider when writing the flight schedule. These regulations are contained in a myriad of instructions governing the operation of U.S. Navy aircraft. Some of the primary instructions include the Naval Aviation Training and Operations Standardization (NATOPS) General Flight and Operating Instruction (OPNAV 3710.7), the NATOPS flight manual for the specific type aircraft, instructions from the squadron's wing, and the squadron's Standard Operating Procedures (SOP). These regulations constrain the number of hours that a pilot, student or instructor, can fly and specify the qualifications that a pilot must hold before flying certain missions or maneuvers. The flight scheduler must ensure the flight schedule does not violate any of these regulations.

b. Maximum number of days a student has to complete the syllabus

A student is ordered to an FRS for a specified time period, generally several months. After completing training at the FRS the student is expected to transfer to an operational squadron. The number of days that a student is assigned to the FRS is a constraint in that it is the maximum time available to complete the student's training without affecting his/her follow-on orders. The flight scheduler must track each student's progress through the syllabus to ensure that he/she will complete the syllabus on time.

c. Events required to be completed before an event is scheduled

Each event has a set of requirements that must be completed before it can be flown. Events in the syllabus build on one another, but are not strictly sequential. Using the USMC Trainee Model[Ref. 3] as an example, a student is required to complete the first five "FAM" (familiarization) flights sequentially. After this, the flight scheduler has the flexibility to schedule the student for the next FAM flight, or an "INS" (instrument) flight. Keeping track of an event's prerequisite requirements is another task for the flight scheduler.

d. Flight hours available for the day

The number of flight hours available per day is constrained by the number of flight hours assigned to the squadron by the squadron's wing. The squadron has the flexibility to determine the number of hours to fly on a given day, but the total number of flight hours for a month, or quarter cannot exceed the squadron's allocation. The monthly or quarterly total allocation of flight hours is generally broken down into weekly or even daily flight hour goals that can constrain the daily flight schedule.

e. Number of aircraft available for the day

A squadron has a fixed number of aircraft. On a given day, only a certain number of aircraft will be available for training flights. When the number of syllabus flight hours exceeds the number of aircraft flight hours available, this constraint becomes binding.

f. Sunrise, sunset and other constraints

Most of the flights in the FRS take place during daylight hours. This presents a constraint in that the flights can only be conducted between sunrise and sunset. Conversely, flights that require nighttime are constrained in that they can only be flown between sunset and sunrise. Other constraints imposed by the operating environment are airport operating hours and the availability of limited resources such as weapons ranges and ships for deck landing qualifications.

3. Problem description summary

The flight scheduler has a great deal of information in the form of inputs and constraints that must be maintained and updated on a daily basis prior to writing the flight schedule. As an example, the scheduler must track and update each student's progress through the syllabus on a daily basis, recording the events that the student has completed as well as noting the events that the student is now qualified to fly. For instructors, the scheduler must track each instructor's total hours flown, qualifications held to teach events, and qualifications that have expired. Maintaining this data is a tedious, but vital task for the flight scheduler. Maintaining a list of the constraints on the flight schedule is also a requirement for the flight scheduler. Constraints on the schedule are more static than the inputs. With the exception of flight hours available to fly on a given day, constraints can be considered to be fixed, and are often imposed as sets of rules.

C. STEPS IN THE FLIGHT SCHEDULE PROCESS

1. Determine mission requirements for the day

The flight scheduler at a FRS determines the mission requirements for the flight schedule from a projection of weekly training requirements. The weekly requirements come from a training matrix designed to ensure that students complete the syllabus in the prescribed time period. The training matrix does not account for factors such as periods of aircraft unavailability or bad weather that might prevent the weekly training goals from being met.

In the FRS, a mission is defined as a student and the event that he/she must

complete. In preparing the set of missions for a given day, the scheduler must first determine which students are available to fly and which events the students needs to complete. Selecting students who are available for the flight schedule is a trivial process. Deciding which events each student is qualified to fly is not so trivial. For each student, the scheduler must first determine which events the student needs to complete. Second, the scheduler must ensure that all pre-requisite flights for the student's needed events have been completed prior to pairing the student to the event. Third, the scheduler must ensure that the student meets all regulations to fly the assigned event. An example of this type of regulation is the requirement that a student must have completed a night flight no more than seven days prior to flying a night deck landing qualification (DLQ) event.

2. Flight hour availability

Once a set of possible missions is selected, the flight scheduler must start adding constraints that may restrict the flight schedule. The first constraint is the number of flight hours available. The sum of the hours required to fly the proposed flight schedule must be less than or equal to the flight hours available on a given day. If the required hours is greater than available hours, missions must be cut.

3. Number of aircraft available

The number of aircraft available for a given day, which can be measured in available aircraft hours, varies due to aircraft maintenance requirements. For the time horizon of a daily schedule, this can be considered to be fixed. The sum of flight hours for the proposed missions must be less than or equal to the number of aircraft hours available. As with flight hour availability, if the proposed flight schedule is greater than the number of available aircraft hours, the schedule must be cut.

4. Assigning missions to aircraft

Once the number of aircraft available is determined, missions are matched to aircraft. Assigning missions to aircraft generally starts by determining sunset and

scheduling flights that require nighttime after sunset and flights that require daylight before sunset.

5. Assignment of instructors to events

The final step in the flight scheduling process for an FRS is to assign instructors to events and aircraft. Instructors are assigned to flights based on several criteria. The primary criterion in assigning an instructor to a student/event pair is the instructor's qualification to instruct the event. The secondary criterion used is based on the number of flight hours that the instructor has flown in the past. In general, the scheduler tries to ensure that flight hours are distributed equally among all instructors. Other criterion, such as an instructor's teaching style may also be considered by the scheduler when pairing instructors to students.

D. CURRENT TECHNOLOGY

Flight schedules are currently written by hand with aids such as a magnetic white-board, magnets with each pilot's name embossed on them and a grease marker. With the exception of word processors to print the schedule and spreadsheets to track pilot flight hours and qualifications, computers have little impact on the flight scheduling process.

Given the large amount of data that the scheduler must update and maintain on a daily basis, automation of the flight scheduling process at military squadrons is long overdue. Next, several approaches to the automating the flight schedule process will be presented.

E. ALTERNATIVE METHODS

Many options have been explored to automate the flight scheduling process. Some companies in the commercial airline industry use mathematical models based on set partitioning to optimize their use of aircrews and aircraft[Ref. 6]. The Navy Test Pilot School has used a model based on heuristic logic to develop a "satisficing" flight schedule[Ref. 7]. Work at the Naval Postgraduate School in the field includes a

mathematical model for a Marine Corp helicopter fleet replacement squadron by Kawakami[Ref. 3].

These solutions have two common characteristics. Each proposes to improve the flight scheduling process with an automated system. Each also sees a need for a database to store the knowledge about the inputs to and constraints on the flight schedule process.

F. OPTIMIZATION APPROACH TO THE FLIGHT SCHEDULE PROCESS

Kawakami models the FRS flight scheduling problem as an integer program. The objective of his model is to schedule students to ensure that they complete the syllabus within the time (in days) allowed. A mathematical description of this model is provided in Appendix A. The GAMS data model and output control statements are provided in Appendix B. While this approach appears to provide a good solution to the flight scheduling process, it has several drawbacks stemming from GAMS' crude user interface and poor data modeling capabilities. To correct for these deficiencies, the GAMS model will be embedded into a DSS and links created to a user interface and relational database. The specific problems caused by the GAMS user interface and data model are listed below.

1. Model integrity

The General Algebraic Modeling System (GAMS) model integrates the data used in the model with the model itself. Every time the data is modified, the scheduler runs the risk of accidentally modifying the model. This means that the model integrity cannot be guaranteed.

2. Data integrity

Data stored in a GAMS model is polyinstantiated, meaning that attributes of an instance of an element are spread over several tables. In the GAMS data model, a table, or relation is created for each set, parameter and table. This makes modification of GAMS data difficult, time-consuming and leads to the problem of "modification anomalies,"[Ref. 9] i.e., data inconsistencies.

3. Input of data

Because of polyinstantiation, input of data into the GAMS model is also difficult. To add an item to a set, first the item of the set must be created, followed by the creation of all associated parameters and tables. Because data structures are only weakly enforced in the GAMS model, the user can not be sure that each parameter and table is correctly populated until the GAMS program is compiled. Because the input of data is so time-consuming, the number of modifications that can be made to a GAMS model on a given day is limited.

4. Management of multiple input and output sets

Using a given GAMS model, multiple solution sets cannot be stored without saving a copy of the integrated GAMS data and model. In addition to the added overhead of storing multiple copies of the model, this problem prevents the automated comparison of solution sets.

5. Friendly presentation of data

While GAMS does have limited report-writing capabilities, the user should be able to quickly and easily change the format of the model output. Using a stand-alone GAMS model, the user cannot change the view of the output without modifying the output control statements and re-compiling the model. It is desirable for the user to be able to format the results of a GAMS model based on the users desires, not the capabilities built into the GAMS model.

III. DECISION SUPPORT SYSTEM APPROACH

This chapter describes the method used to integrate the GAMS flight scheduling model into a DSS and the advantages of doing this. First, a comparison of the flight scheduling process using a stand-alone GAMS model will be made to the same process with that model embedded in a DSS. This shows some of the advantages of the DSS approach. Next, the four steps used to integrate a GAMS model into a DSS will be discussed. The chapter concludes with a detailed discussion of the advantages of using a DSS to solve the flight scheduling problem.

A. A COMPARISON OF THE GAMS AND DSS APPROACH

In Chapter II, two problems with the manual flight scheduling process were discussed, (1) lack of automated ways to store information (both input data and constraints) needed to write the flight schedule, and (2) a lack of objective criteria to help the scheduler determine if a flight schedule makes maximum use of the squadron's aircrew and aircraft. The GAMS approach solves the second problem by placing the variables and constraints used in the flight scheduling process into a mathematical model and then solving for the optimal values of the variables. What the stand-alone GAMS model lacks is a way to quickly capture the data required by the model. To show how integrating the GAMS model into a DSS can simplify the flight scheduling process a comparison between the GAMS and DSS approach is provided.

1. The GAMS approach

Using the GAMS model a flight schedule is produced using the following methodology: 1) Determine what instructors and students are available for inclusion into the flight schedule. 2) Edit the model's sets, parameters and tables to reflect instructor and student availability. Instructors and students who are added to, or deleted from a set must also be added to, or deleted from each parameter and table where they are represented. Figure 1 demonstrates that to add a single student to the GAMS model, the student must be added to five different data fields. 3) Modify the

```
A. Original data model
SETS
  P  student pilots (11)
      / DARLING, SHEERIN, . . ., KANG /
PARAMETERS
  IBAR(P) max training items per day for pilot P
       / DARLING 2, SHEERIN 2, . . ., KANG 2  /
  DHAT(P) Number of days pilot P is assigned for training
        / DARLING 42, SHEERIN 42, . . ., KANG 7  /
  NC(P) Number of finished items for pilot P
        / DARLING 11, SHEERIN 12, . . ., KANG 1  /
TABLE
  PROG(I,P) completed items I for student P
       DARLING SHEERIN . . .         KANG
FAM100   1       1                 1
FAM101   1       1                 0
  .      .       .                 .
  .      .       .                 .
  .      .       .                 .
CCX190   0       0 . . .           0

B. Modified data model to include the student "JONES"
SETS
  P  student pilots (12)
      / DARLING, SHEERIN, . . ., KANG, JONES /
PARAMETERS
  IBAR(P) max training items per day for pilot P
       / DARLING 2, SHEERIN 2, . . ., KANG 2, JONES 2  /
  DHAT(P) Number of days pilot P is assigned for training
       / DARLING 42, SHEERIN 42, . . ., KANG 7, JONES 0 /
  NC(P) Number of finished items for pilot P
       / DARLING 11, SHEERIN 12, . . ., KANG 1, JONES 0  /
TABLE
  PROG(I,P) completed items I for student P
       DARLING SHEERIN . . .         KANG JONES
FAM100   1       1                 1    0
FAM101   1       1                 0    0
  .      .       .                 .    .
  .      .       .                 .    .
  .      .       .                 .    .
CCX190   0       0 . . .           0    0
```

Figure 1. Modifications required to add a student to the GAMS model.

parameters to reflect current values. Each parameter must be reviewed to determine if any changes in the parameter need to be made. An example of this type of change in the USMC Trainee Model is the parameter DHAT(P) that represents the number of days that a pilot P has been assigned for training. Each day, the scheduler must increment DHAT(P) for all students. 4) Run the GAMS model. 5) Read the GAMS solution. The primary advantage of using a GAMS model over manual scheduling is that it provides the scheduler a method of objectively determining which instructor/student pairs should be assigned to the flight schedule.

2. The DSS approach

The primary advantage of embedding the GAMS model in a DSS is that the data needed by the model can be stored and maintained separately from the model. This allows the user to modify the model's data more quickly and easily than if it were stored in the GAMS model itself.

Using a complete DSS the flight schedule can be produced using the following methodology: 1) Determine what set of instructors and students are available for inclusion into the flight schedule. Instead of manually adding or deleting individuals from the GAMS data fields, a pull-down menu listing the instructors and students is provided to the user. Each person (instructor or student) can be included to the schedule by marking them for inclusion in the schedule. The design of the database, which will be discussed later in this chapter, eliminates the requirement to modify a set's associated parameters and tables (step two in the GAMS approach). 2) Modify the parameters to reflect current values. The scheduler may desire to change some parameters; this can be done through pull-down menus. Other parameters such as DHAT(P) can be automated to increment on a daily basis. This requirement in step three of the GAMS approach can be automated and thus eliminated. 4) Invoke a DSS command that exports database data to the GAMS model, executes the GAMS model and imports the results of the GAMS model back into the database. 5) Read the results of the GAMS model on customized reports.

3. Advantages of the DSS over the GAMS approach

Using the DSS approach, the process of generating the flight schedule is significantly simplified. Because all data required by the GAMS model is stored in a database, with its built-in capabilities to handle inclusion and exclusion of set members through Structured Query Language (SQL) queries, modifications of the data sets are significantly simplified. Parameters, such as DHAT(P) that change on a daily basis, can be automated to self increment based on the current date and the student's start date. The DSS also provides a method for the user to select the format of the solution presentation, making the interpretation of the GAMS results more user-friendly.

There are four primary steps required to integrate a GAMS model into a DSS: 1) Create an external database to store the data needed by the GAMS model. 2) Create a set of linkages from the database to the GAMS model. 3) Create a set of linkages from the GAMS model to the database. 4) Create a user interface that provides a seamless and user-friendly interface for the DSS.

B. DESIGN OF THE DATABASE

An entity-relationship diagram depicting the data structures used in the USMC Trainee Model is provided in Appendix C. This diagram depicts 11 entities. Each of these entities represents a GAMS set, parameter or table. The problem with this data structure is that the attributes of a single GAMS set are spread over several entities. This type of data dispersion is known as "polyinstantiation" and leads to the problem of "modification anomalies." To illustrate this problem, consider the set P, and parameters IBAR(P), DHAT(P), and NC(P). If a member of the set P is deleted, this change is not reflected in the parameters IBAR(P), DHAT(P) and NC(P). When designing a database to support a GAMS model the goal is to eliminate the problem of modification anomalies caused by polyinstantiation. Before the database design is discussed, an overview of relevant relational database terms and concepts is presented.

1. An overview of relational database concepts

A relational database organizes and represents data in the form of relations. A relation is a two-dimensional table consisting of rows and columns. Each row represents a record while each column represent an attribute of the relation. Records in the relation are identified by a group of one or more unique attributes known as a "key." A key that is "an attribute of one or more relations other than the one in which it appears"[Ref. 9] is known as a "foreign key."

2. Selection of relations from GAMS sets

When designing a database to support a GAMS model, each mathematical set becomes a relation. The USMC Trainee Model (Appendix B), has three sets: I, items of the syllabus, P, student pilots, and Q, instructors. These three sets become the relations I, P, and Q in the database. The key for relation I is the event name, the key for relation P is the student's last name, the key for relation Q is the instructor's last name.

3. Selection of attributes from GAMS parameters

Attributes of each relation are obtained from the parameters of the mathematical model. Each parameter becomes an attribute of the relation it describes. Using the USMC Trainee Model as an example, set P has three parameters: IBAR(P), the maximum number of training items per day for pilot P, DHAT(P), the number of days that pilot P has been assigned for training, and NC(P), the number of finished items since assignment of pilot P to the squadron. These three parameters are included in the relation P. The relation P now consists of four fields, one key field and three attributes IBAR, DHAT and NC. The composition of the relation P and its correspondence to the four GAMS tables it represents is shown in Figure 2.

4. Selection of relations from GAMS tables

A GAMS table defines an attribute that is dependent on two or more sets. While the structure of a GAMS table and a relation differ, they convey the same information. The GAMS table QUAL(I,Q) denotes a single attribute, QUAL, that is

dependent on two GAMS sets, an instructor Q and event I. This same information can be represented in a relation where the instructor Q and event I are represented as foreign keys with a single attribute QUAL. The USMC Trainee Model has three tables that will become three relations in the relational database.

Object P		Mathematical Model	
Attribute Name	Type	Attribute Name	Type
Last Name	(Key Field)	P	Set
IBAR	(Attribute)	IBAR(P)	Parameter
DHAT	(Attribute)	DHAT(P)	Parameter
NC	(Attribute)	NC(P)	Parameter

Figure 2. Relationship between object P and GAMS data.

While only two-dimensional tables are addressed in this thesis, three or more dimensional tables can be represented by a relational database using the same logic presented above. Suppose a modeler desired to track an aircraft's position over time. To do this, he/she would need a four-dimensional GAMS table that might have the format POSITION(LAT, LONG, ALT, TIME). This information can be represented in a relation where LAT, LONG, ALT, and TIME all become foreign keys with an attribute POSITION, which is dependent on all four keys.

5. USMC Trainee Model database design

An entity-relationship diagram of the database designed to support the USMC Trainee Model is provided in Appendix D. The eleven entities used to store data in the GAMS model are combined into six relations. The six relations, depicted as entities in the diagram, represent the three GAMS sets and three GAMS tables used in the USMC Trainee Model. It should be noted that the five GAMS parameters were incorporated into the relation they describe. The lines connecting the entities show the relationships between the entities. These relationships are enforced through the use of keys and foreign keys.

The data model shown in Appendix D is in the "third normal form."[Ref. 9] Relations are considered in the third normal form if: 1) all non-key attributes are dependent on all of the key, and 2) no transitive dependencies exist[Ref. 9]. The significance of having a simple data model in the third normal form is that it is free of modification anomalies.

C. LINKS FROM THE DATABASE TO THE MODEL

In a stand-alone GAMS model, the data and the model are both contained in a single computer file. This can lead to a model integrity problem as discussed in Chapter II. A DSS solves this problem by physically separating the data from the model. Because the data and model are now separate, a set of links must be created to import database data into the model at run time. The process used to create the links between the database and GAMS model is described below.

1. Determine the data requirements of the model

Each link between the database and mathematical model consists of a database query and a conversion of the query data to a file that is readable by the GAMS model. Each GAMS set, parameter and table will require a link. The USMC Trainee Model has 11 sets, parameters and tables, leading to a requirement for 11 links. Scalars are single value inputs in a GAMS model and can be treated as an input requiring a link, entered by the user at model run time, or can be left in the model as fixed values. In this thesis, scalars are treated as being entered at run time, and as fixed values requiring no links.

2. Determine required database queries

Queries are used to extract the required data from the database. Queries for sets will extract only the key field from the database relation being queried. Queries for parameters will extract the key field and the corresponding attribute of the relation being queried. Queries for tables will extract the two foreign keys and the attribute of the relation being queried. A complete listing of the USMC Trainee Model required inputs and supporting database query names is provided in Appendix E.

Appendix F displays the Structured Query Language (SQL) queries used to extract the required database data.

3. Conversion of database data to mathematical model inputs.

The results of the queries are stored as database relations. To export the query data, it must first be converted to a format readable by the GAMS model. Next, it must be placed in a file that is accessible by the mathematical model. Using a Paradox to GAMS link, each query result is first converted into a separate ASCII file, given a name to reflect the GAMS input it contains, and stored in a location where it is accessible by the mathematical model. When the mathematical model is run, these files are imported into the model. GAMS provides a $include statement that is used to import data into the model. The original USMC Trainee Model modified to import Paradox data is provided in Appendix G. As can be seen, each of the model's sets, parameters and tables is replaced with a $include statement followed by a file name.

D. LINKS FROM THE MODELBASE TO THE DATABASE

GAMS output capabilities are limited. Using the DSS approach, this problem is overcome by importing the GAMS solution sets into the database. This is done through a set of links from the mathematical model to the database. A link is needed for each GAMS solution of interest to the user. Each link consists of a GAMS script that stores the solution in a file accessible to the database and a database script to import the file into the database.

The original USMC Trainee Model shown in Appendix A uses the "DISPLAY" function to produce the solution. To send the solution set to an output file a script similar to the one shown in Figure 3 is used.

Line (1) of the GAMS script creates an alias "TRN" for the path "C:\FLIGHT\TRAINING.TXT." Line (2) is a command to send the model output to the alias. Line (3) is a print control command to produce the solution set in a text

quoted, comma delimited format. Line (4) converts the solution from a GAMS table format, to a format that lists each attribute of the output by column.

```
FILE TRN /'C:\FLIGHT\TRAINING.TXT'/;                  (1)
PUT TRN;                                              (2)
TRN.PC=5;                                             (3)
LOOP ( IP (I,P)$(X.L(I,P) ), PUT P.TL, I.TL /;);      (4)
```

Figure 3. GAMS solution set export script.

The GAMS script shown above will produce an ASCII file that is accessible by the database. A script must be written in the database to convert the ASCII file into a relation that can be read by the database. Figure 4 displays a Paradox-specific script used to import the results of the GAMS model. The code presented in Figure 4 is a command to convert the text-quoted, comma-delimited ASCII file "teacher.txt" into the database relation "teacher.db."

```
importASCIIvar("teacher.txt", "teacher.db",
        "," , "" , True, True)
```

Figure 4. Paradox importASCIIvar script.

E. DESIGN OF THE DIALOGUE GENERATOR

The dialogue generator provides a user-friendly interface between the user and DSS while controlling the user's access to the components of the DSS. For the prototype DSS developed for this thesis, the dialogue generator was created using the graphic user interface (GUI) features provided by Paradox. The interface allows the user to have access to data and functions that are needed, but prevents the user from modifying data and functions that should not be changed by the user.

The DSS for the USMC Trainee Model uses a hierarchical menu structure. The user is presented a main menu that can be traversed by the type of functionality

desired. This type of interface allows the user to "drill-down" from a very general menu to more specific menus that offer increased functionality. Because the user is only given access to functions that are presented on the menu, functions that should not be accessed by the user are hidden. Appendix H provides an overview of the menu structure used in the USMC Trainee Model. The top diagram shows the functions provided by the main menu. The three lower diagrams show the functionality provided at the second level of the hierarchy by the "Add Menu," "Modify Menu," and "Delete Menu" buttons.

The main menu allows the user to add, modify, or delete an item in the database, run the GAMS model, or view the GAMS solution. The user selects the desired functionality through a GUI button. Selecting one of these buttons will take the user to the next level down on the menu structure. As an example, the second level of the menu hierarchy for the "ADD" function allows the user to add an instructor, student or event, or to return to the main menu. Using a hierarchical menu structure provides the user with several functional choices, grouped by functionality, without providing too many choices as to be confusing, or cluttering the screen. This structure works well for a DSS with limited capabilities. Larger decision support systems would probably need to employ a more dynamic user interface to prevent the hierarchy of menu from becoming too deep.

1. Data entry

If the user desires to enter a new instance of a student, he/she traverses the menu hierarchy to the "Add a Student" menu. A form is displayed on the screen that provides all of the fields that require data. The new instance of the student will not be accepted by the database until each required field is filled in. This requirement ensures that all of the data needed by the model resides in the database. Once a complete set of data is entered, the dialogue generator is designed to ensure that the data is saved in the proper tables, thus avoiding the problem of modification anomalies.

2. Data modification

Data modification is completed through the dialogue generator. If a user desires to modify a parameter of a mathematical set, the user calls up the modify form. The user identifies the instance of the set to be modified and is provided access to fields where modifications are allowed. The set identifier, i.e., an instructor, student, or event name can not be modified by the user. These set identifiers are key fields in the database. A change to one of these identifiers would be interpreted as the creation of a new element of the set.

There are fields other than the key field where it might be desirable to restrict write access when a record is modified. An example of this is a student's start date. By preventing write access to this field, the scheduler can never accidentally change this field.

3. Data deletion

The removal of data from the database is also done at the dialogue generator level. This is preferable to having the user remove data directly from the database relations in that it ensures that when a set is removed, it is removed not only from its primary table, but also from any other tables where the element's key is used as a foreign key.

4. Bounds checking

An input to the database can be checked to ensure that it is within an acceptable range before it is entered into the database. This is completed through the assignment of an upper and lower allowable limit that the data must pass before being forwarded to the database. An example of such a bound check in the USMC Trainee Model is for the parameter HBAR(Q), the maximum flight time per day for an instructor Q. The squadron might have a rule that states that no instructor will fly more than six hours a day. This rule can be enforced at the dialogue generator by preventing any instructor from having an HBAR(Q) value greater than 6.0 hours. Each time that a record for an instructor is added or modified, the bounds checker

will ensure that HBAR(Q) is within allowable limits or the change to the database will not be allowed.

F. ADVANTAGES

Integrating a mathematical model into a DSS provides many advantages over the use of a stand-alone model. Mathematical models are designed to facilitate model representation and execution but lack a formal data model and management system. To overcome this deficiency, the USMC Trainee Model was integrated into a DSS. Benefits of this approach include explicit data modeling, improved representation and manipulation of model inputs and outputs, greater integrity of input data, and easier interpretation of multiple views of model outputs. This section will discuss these benefits in detail and how they can make the life of the scheduler easier.

1. Model integrity

The interaction between the user and the mathematical model is through the dialogue generator. The dialogue generator allows the user to modify the data used by the mathematical model, but not the model itself. Because the user does not have access to the mathematical model, the possibility of accidentally modifying the mathematical model while modifying data is eliminated. The integrity of the model can now be guaranteed because there is no direct user access to the model.

2. Input of data

Using a mathematical model, data is entered in a format that is recognizable to the program. Using a DSS, data input screens can be created that take data from the user in formats that are recognizable to humans.

Using a stand alone GAMS model, each time an element is added to a set, all associated parameters and tables must be modified to reflect this change. Figure 1 provided an example of this problem. In the USMC Trainee Model, each time a student is added to a set, he must also be added to three parameters and one table, PROG(I,P). Because the table PROG(I,P) denotes a student's progress through the 34 events in the syllabus, each time a student is added to the GAMS model, 38

modifications must be made to the data portion of the model. Using a DSS, a single screen can be created that prompts the user for all of the required information to enter a new student. Once this data is entered, the new student is entered into all of the appropriate sets, parameters and tables with only a single data entry.

3. Data integrity

Use of a DSS also ensures that data integrity is maintained when a record is modified. To delete a student directly from a mathematical model requires that the student be removed from all appropriate sets, parameters and tables. Using a DSS, the user traverses the menu hierarchy to the delete-student menu. In the delete-student menu, the student's name is either entered or selected from a drop-down list of names and the instance of the student is automatically deleted from all database tables.

Another type of data integrity problem encountered when using the mathematical model directly is the problem of using different spellings to represent the same index of a set, parameter or table. Using a mathematical model directly, the scheduler might identify a student in the set students as "Smith," and then in a parameter or table accidentally label the same student as "Smyth." With a mathematical model, this error will not be detected until the model refuses to run because of improper data. Using the relational database, the student's name is only entered once, so either Smith or Smyth would be acceptable, but the problem of using different spellings in the same model is eliminated.

4. Automated modification of dynamic data

Dynamic parameter and table values can be automatically generated through queries of the database. In the USMC Trainee Model, the parameter DHAT(P), which represents the number of days that a student P has been available for training, needs to be manually updated daily. Using the DSS, this parameter can now be automatically determined thought the expression DHAT(P) = TODAY - Start-Date.

5. Alternative views of data

The use of a DSS supports multiple views of the mathematical model's results. Once the solution set is extracted from the mathematical model, it can be presented to the user as text, graphs, or charts, depending on the user's preferences.

6. Management of multiple input and output data sets

Using a stand-alone mathematical model, the input data, mathematical model and solution set are all stored in a single output file. Using the DSS approach, output data is stored in the database where it can be readily accessed for comparison with other solution sets, or used as an input to other models.

Another advantage of storing data separate from the model is that if the user is not happy with the solution set, he/she can make changes to the data and quickly re-run the model. As an example, in the USMC Trainee Model DSS, drop-down menus are provided that allow the user to change HBAR(Q), an instructor's maximum flight hours per day and IBAR(P), a student's maximum training items per day. If the scheduler is unhappy with a solution, he/she can change these parameters in the drop-down menu and re-run the model, all within a few minutes. This time savings allows the user to run and store many scenarios. Once the different solution sets have been stored, their inputs and outputs can be compared against one another.

7. Friendly presentation of solutions

A DSS provides the user with the ability to quickly and easily modify the output of the GAMS solutions. Once a solution is imported into the database it can be modified to display what is important to the user. Different users may have different needs for the results of a mathematical model. A senior manager may only be interested in an executive summary, while the flight scheduler might be interested in a much finer level of detail. Different reports can be created for the same data using the dialogue generator.

G. SUMMARY

This chapter has shown that the DSS approach to the flight scheduling problem has many advantages over the use of a stand-alone GAMS model. The advantage most critical to the user is that the task of writing the flight schedule on a day-to-day basis is greatly simplified. Another advantage is that access to a formal data model and management system is provided when the mathematical model is linked with a database.

The steps used to integrate a mathematical model into a DSS were also shown. The steps include (1) design of the database to support the mathematical model, (2) creation of a set of links from the database to the mathematical model, (3) creation of a set of links from the mathematical model to the database, and (4) creation of a user interface that is user friendly while restricting access to components of the DSS that should not be manipulated by the user.

IV. AUTOMATION OF THE DSS APPROACH

In Chapter III the process used to integrate a GAMS model into a DSS was discussed. In this chapter, a set of generalized procedures is presented that could be used to integrate any GAMS model with a database. In theory, these generalized procedures can be built into a software package that will allow the GAMS model user to automatically generate a database capable of supporting a GAMS model. While prototyping of some of the required modules has been done, a fully functioning software package has not been created due to time constraints.

Creating the relations and linkages required to integrate Kawakami's GAMS model into a DSS was a time intensive process. The use of a software package that automates this process will provide individuals with no knowledge of relational databases or mathematical optimization models access to GAMS models. Using this approach, the model user can focus his/her attention on the data used by the model while the inner-workings of the GAMS model remain hidden.

The process described in this chapter entails (1) the formal capture of all GAMS input and output data requirements, (2) the creation of a relational database to support the GAMS model, (3) the creation of queries to extract the proper data for all GAMS data sets, parameters and tables, (4) the transfer of data to the GAMS program, and (5) the transfer of GAMS solutions to the relational database.

A. CAPTURE OF THE GAMS MODEL METADATA

Metadata is information about the structure of the GAMS data that will be stored in the database. The first step in creating an automated process to create the database relations and links is to record every set, parameter, table and output used by the GAMS model. For a specific GAMS model, the metadata information is obtained from the data definition portion of the model. The metadata table consists of four columns, GAMS-NAME, GAMS-TYPE, INDEX1 and INDEX2. GAMS-NAME describes the name of the data set. GAMS-TYPE describes the type of data

held by the set, (SET, PARAMETER, TABLE, or OUTPUT). INDEX1 identifies the GAMS set that a parameter, table or output describes. INDEX2 also describes the GAMS set, but is only used with tables or output types. Figure 5 provides a sample metadata table that represents a portion of the metadata for the USMC Trainee Model. Once captured, the metadata is used to generate the relations, queries and linkages required to integrate a relational database and a GAMS model. The meta-data table shown in Figure 5 is designed to support tables with a maximum of two-dimensions. If the GAMS model required greater than two-dimensional tables, extra indices could be added.

GAMS-NAME	GAMS-TYPE	INDEX1	INDEX2
P	SET		
I	SET		
IBAR	PARAMETER	P	
PROG	TABLE	I	P
TEACHER	OUTPUT	I	P

Figure 5. Portion of the metadata required for the USMC Trainee Model.

B. DATABASE DESIGN

The relations and attributes contained in a database designed to support a GAMS model are found by examining the sets, parameters, and tables of the GAMS model. General rules for determining the relations are (1) each GAMS set becomes a database relation, the set name becomes the relation name and the key field of the relation, (2) each GAMS parameter is made an attribute of the relation that it describes, and (3) GAMS tables become relations comprised of two foreign keys and a single attribute. The GAMS table name becomes the name of the relation. The two foreign keys take on the name of the indices of the GAMS table. The relation's attribute takes the name of the GAMS table.

An automated procedure to create the database is shown in Figure 6. In the first portion of step 1, relations are constructed for GAMS sets. In step 2, attributes are added to these relations. In the second part of step 1, relations are created for GAMS tables. This process will create a set of database relations that support a GAMS model, and that are in the third normal form.

C. CREATION OF QUERIES

Once the GAMS data model is modified into a relational database format, it is no longer directly usable by a GAMS model. In a GAMS data model each set, parameter, and table constitute their own data table or relation. In a relational database, sets and parameters are combined into a single relation. To extract the data needed by the GAMS model, queries of the database must be performed to extract the data needed by the GAMS model.

The extraction of data required by the GAMS model can be completed with SQL "select-from" queries. The data requirements for each GAMS set, parameter and table are summarized in the following rules.

1. Each mathematical model set requires the key field from its corresponding database relation.

2. Each mathematical model parameter requires the key field and an attribute from its corresponding database relation.

3. Each mathematical table requires the two foreign keys and the single attribute from its corresponding database relation.

The queries required to retrieve input data for the model can be generated from the metadata table. Each GAMS-TYPE other than type "OUTPUT" will require

```
Step 1

OPEN TABLE METADATA
WHILE NOT END-OF-TABLE
 READ RECORD
  IF GAMS-TYPE = "SET" THEN    ; CREATE A RELATION FOR
   CREATE TABLE GAMS-NAME      ; EVERY SET
    (GAMS-NAME, CHAR(10),
     PRIMARY KEY (GAMS-NAME))
  ENDIF
  IF GAMS-TYPE = "TABLE" THEN  ; CREATE A RELATION FOR
   CREATE TABLE GAMS-NAME      ; EVERY TABLE
    (INDEX1, CHAR(10),
     INDEX2, CHAR(10),
     GAMS-NAME, FLOAT,
     FOREIGN KEY (INDEX1), FOREIGN KEY (INDEX2))
  ENDIF
 GET NEXT RECORD
ENDWHILE
CLOSE TABLE METADATA

Step 2

OPEN TABLE METADATA
WHILE NOT END-OF-TABLE
 READ RECORD
  IF GAMS-TYPE = "PARAMETER" THEN    ; ALTER THE RELATION
   ALTER TABLE INDEX1 ADD GAMS-NAME, FLOAT
  ENDIF
 GET NEXT RECORD
ENDWHILE
CLOSE TABLE METADATA
```

Figure 6. Database design procedure.

a query to extract data for the GAMS model. Figure 7 describes the procedure that is used to generate the required queries. Once generated, the queries can be placed in an export script that is run prior to the running of the GAMS model.

```
OPEN TABLE METADATA
WHILE NOT END-OF-TABLE
  READ RECORD
  IF GAMS-TYPE = "SET"
    SELECT DISTINCT GAMS-NAME
    FROM GAMS-NAME+.DB
    SAVE AS GAMS-NAME+Q.DB
  ENDIF
  IF GAMS-TYPE = "PARAMETER"
    SELECT DISTINCT INDEX1, GAMS-NAME
    FROM INDEX1+.DB
    SAVE AS INDEX1+Q.DB
  ENDIF
  IF GAMS-TYPE = "TABLE"
    SELECT DISTINCT INDEX1, INDEX2, GAMS-NAME
    FROM GAMS-NAME+.DB
    SAVE AS GAMS-NAME+Q.DB
  ENDIF
ENDWHILE
CLOSE TABLE METADATA
```

Figure 7. Query Generation.

D. LINKS FROM THE DATABASE TO THE MODELBASE

Using the DSS approach, data is no longer stored in the mathematical model. This is beneficial in that the model can be validated and then stored so that it cannot be accessed by the user. The drawback is that a linkage must be created between the database and the mathematical model. Two sets of links are needed. One link

provides a path from the database to the mathematical model. The second link provides a path from the model to the database.

Creating the link from the database to the GAMS model entails (1) storing the query data in a format that is readable and accessible by the GAMS program, and (2) importing the data into the mathematical model. Step one is completed by a script that resides in the relational database and can be automated through the use of the data held in the metadata table. A generalized procedure that produces the required links between a Paradox database and a GAMS model is provided in Figure 8.

```
OPEN TABLE METADATA
WHILE NOT END-OF-TABLE
 READ RECORD
 WRITE "ExportASCIIvar("GAMS-NAME+Q.DB",
  "GAMS-NAME+Q.TXT", "QUERY-RESULT-STRUCT.DB", TRUE)"
ENDWHILE
```

Figure 8. Paradox script required to create a Paradox to GAMS link.

Step two is completed by replacing each GAMS set, parameter and table with the script shown in Figure 9.

```
"$include "operating system path\GAMS-NAME+.TXT"
```

Figure 9. GAMS script required to create a Paradox to GAMS link.

E. LINKS FROM THE MODELBASE TO THE DATABASE

For reasons described in Chapter III, it is desirable to store the results of the GAMS model solutions in the database. Creation of links from the GAMS model to the database is a two-step process. In step one, each GAMS "display" statement is replaced with a statement having the structure shown in Figure 10. A description of the meaning of this code is provided in Chapter III.

After the GAMS model is run with the modifications shown in Figure 10, a file is created that contains the solution for each output set identified in the metadata table. Step two is to import the file into the database and convert it into a format that

```
FILE ALIAS-NAME/"OPERATING SYSTEM PATH\GAMS-NAME+.TXT/"
PUT ALIAS-NAME
ALIAS-NAME+.PC=5;
LOOP(INDEX1 INDEX2 (INDEX1 INDEX2)$X.L(INDEX1 INDEX2))
  PUT INDEX2+.TL INDEX1+.TL/;);
```

Figure 10. GAMS export command.

is readable by the database. A procedure to produce the Paradox specific commands required to import an ASCII file and store it as a Paradox table is provided in Figure 11. A full explanation of the meaning of the Paradox code is provided in Chapter III.

```
OPEN TABLE METADATA
WHILE NOT END-OF-TABLE
  READ RECORD
  WRITE "importASCIIvar("GAMS-NAME+.TXT" , "GAMS-NAME+.DB",
       "," , " ", TRUE, TRUE);"
ENDWHILE
```

Figure 11. ObjectPal importASCIIvar script.

F. SUMMARY

The procedure listed above uses the metadata table to create a relational database in the third normal form. Conceptually, the software package will create the relations required to support the GAMS model, and an export and import script. The export script will contain the code required to query the database relations and build the files that will be accessed by the GAMS model. The import script will contain the code required to bring the GAMS solutions into the relational database.

A program will be required to control the execution of the import and export scripts as well as to execute the GAMS model. While the details of this program will not be presented, in its simplest form it would contain three calls, (1) a call to execute the export script, (2) a call to execute the GAMS model, and (3) a call to execute the import script.

V. CONCLUSION AND RECOMMENDATIONS

This paper has shown the advantages of using a DSS over both the manual scheduling of flights and the use of a stand-alone mathematical optimization model. The relationship between relational database models and GAMS data models has also been demonstrated. This relationship can be exploited to integrate a GAMS model into the framework of a DSS.

While the database created provides a linkage from the mathematical optimization model to both the database and the dialogue generator, to be a truly useful DSS more functionality needs to be included. The model created in this thesis only tracks the data required for the USMC Trainee Model. In addition to this data, a working version of this DSS should track data about the flights flown by both the students and instructors. This can be easily accomplished through the creation of a relation that contains flight data and is linked to the pilot and instructor relations.

Flight data is currently captured by the NALCOMIS database at the completion of each flight. At the completion of a flight, the aircraft commander records the flight crew, aircraft launch time, aircraft land time, number of landings made by each pilot, number of approaches made by each pilot and comment codes for the flight. Currently this data is not used by the squadron scheduler. Instead, data required by the scheduler is taken from paper copies of the NALCOMIS reports. One way to greatly enhance a flight-scheduling DSS would be to create a set of SQL commands that would import the NALCOMIS data directly into the flight scheduler's DSS.

Chapter IV discussed the generalized theory used to integrate a GAMS model and a relational database. These generalized procedures, based on the use of a metadata table to store information about the structure of a GAMS model's data can be used to create a software package that will automatically create the required database relations and linkages required to integrate a GAMS model into the framework of a DSS. While this automated procedure was not implemented, its

development will allow individuals with no understanding of the GAMS language access to its powerful capabilities.

LIST OF REFERENCES

1. Sprague, R.H., "A Framework for the Development of Decision Support Systems," *MIS Quartery*, pp. 1-27, December 1980.

2. Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User's Guide*, The Scientific Press, 1988.

3. Kawakami, T., *An Aid for Flight Squadron Scheduling*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1990.

4. Bhargava, H.K., Krishnan, R., and Mukherjee, S., "*On the Integration of Data and Mathematical Modeling Languages*," Annals of Operations Research, v. 38, pp. 69-95, 1992.

5. Krishnan, *Knowledge Based Aids For Model Construction*, Ph.D Thesis, University of Texas at Austin, 1987.

6. Marsten, R.E. and Shepardson, F., "Exact Solution to Crew Scheduling Problems Using the Set Partitioning Model; Recent Successful Applications," *Networks*, v. 11, pp. 165-177, 2 January 1981.

7. Davis, M.W., "A Resource Assignment and Management Information System for Event Scheduling in a Flight Training Environment," *Interfaces*, v. 10, pp. 68-73, 4 August 1980.

8. Dewitz, S. and Olson, M., *Semantic Object Modeling with Salsa, A Tutorial*, Mitchell McGraw-Hill, 1994.

9. Kroenke, D.M., *Database Processing: Fundamentals, Design, Implementation*, Macmillan Publishing Company, 1992.

BIBLIOGRAPHY

Borland International, *Paradox for Windows User's Guide*, Scotts Valley, California, 1985.

Borland International, *Paradox for Windows Guide to ObjectPal*, Scotts Valley, California, 1985.

APPENDIX A. MATHEMATICAL DESCRIPTION OF USMC TRAINEE MODEL

Mathematical description from Ref. 3

1. Index Sets

$t \in T$	current date
$p \in P_t$	pilots (student pilots) available for training on day t
$q \in Q_t$	instructors available for teaching on day t
$q \in Q_t^i$	set of instructors who are qualified to teach item i
$i \in I$	training flights (items in syllabus)
$i \in I_{pt}^O$	unfinished items with exactly one prerequisite remaining for pilot p
$i \in I_{pt} = I_{pt}^O \cup I_{pt}^1$	set of unfinished and potentially allowable items
$(i,i') \in II_{pt} = \{(i,i') \mid i \in I_{pt}^O,\ i' \in I_{pt}^1, i < i'\}$	set of pairs of items that are allowable and such that one is a prerequisite of the other,
$i \in I_F^D$	daytime formation items and
$i \in I_F^N$	nighttime formation items (actually only a single item).

2. Data

H_t	flight hours goal on a day t,
h_i	flight hours needed for syllabus item i,
$\bar{I}_p$	maximum number of training items per day for pilot p,
$\bar{H}_q$	maximum number of flight hours per day for instructor q,
$\hat{N}_{pt}$	current goal for items to be completed by pilot p,
N_{pt}	number of items completed by pilot p,
C'	objective value for penalty variable Z^+ and Z^-,

C^1_{pt} = $(1+[\max\{0, \hat{N}_{pt}-N_{pt}\}]^2)$,

C^2_{pt} = $(1+[\max\{0, \hat{N}_{pt}-(N_{pt}+1)\}]^2)$ and

C'_{pt} = $(C^2_{pt}-C^1_{pt})$.

3. Decision Variables

X_{ip}	1 if a student pilot performs item i, 0 otherwise.
Y_{ipq}	1 if an instructor q teaches student p item i, 0 otherwise.
W_p	1 if a student pilot p flies two items on a given day, 0 otherwise.
Z^-	underachievement for flight hours goal,

Z^- overacheivement for flight hours goal and

V, V' non-negative integer variables for pairing formation flights.

4. Formulation

$$\text{Maximze} \quad \sum_{p\in P_t} \sum_{i\in I_{pt}} C^1_{pt} X_{ip} - C'(Z^+ + Z^-) = \sum_{p\in P_t} \acute{C}_{pt} W_p$$

$$\text{Subject to} \quad \sum_{q\in Q^i_t} Y_{ipq} - X_{ip} = 0 \quad i\in I_{pt}, \forall\, p\in P_t \tag{1}$$

$$\sum_{p\in P_t} \sum_{i\in I_{pt}} h_i Y_{ipq} \leq \bar{H}_q \quad \forall\, q\in Q_t \tag{2}$$

$$\sum_{i\in I_{pt}} X_{ip} \leq \bar{I}_p \quad \forall\, p\in P_t \tag{3}$$

$$X_{i'p} - X_{ip} \leq 0 \quad (i,i')\in II_{pt}, \quad \forall\, p\in P_t \tag{4}$$

$$\sum_{i\in I_{pt}} X_{ip} - W_p \leq 1 \quad \forall\, p\in P_t \tag{5}$$

$$\sum_{p \in P_t} \sum_{i \in I_F^D} X_{ip} - 2V = 0 \qquad (6)$$

$$\sum_{p \in P_t} \sum_{i \in I_F^N} X_{ip} - 2V' = 0 \qquad (7)$$

$$X_{ip} + X_{i'p} \leq 1 \qquad i,i' \in I_F^D, \ \forall \ p \in P_t \qquad (8)$$

$$\sum_{p \in P_t} \sum_{i \in I_F^D} Y_{ipq} \leq 1 \quad \forall \ q \in Q_t \qquad (9)$$

$$\sum_{p \in P_t} \sum_{i \in I_{pt}} h_{iX_{ip}} + Z^- - Z^+ = H_i \qquad (10)$$

$$Z^+, Z^- \geq 0$$

$$V, V' \in \{0, 1, 2, ...\}$$

$$X_{ip} \in \{0,1\} \quad \forall \ i,p$$

$$Y_{ipq} \in \{0,1\} \quad \forall \ i,p,q$$

Explanation of constraints:

Constraint (1) assigns exactly one instructor to each item flown.

Constraint (2) limits the number of flight hours for each instructor pilot during the day.

Constraint (3) limits the maximum number of training flights for each pilot during the day.

Constraint (4) ensures that prerequisite items are completed before items requiring prerequisites.

Constraint (5) is used to modify the objective function value if two items are performed by a pilot instead of just one.

Constraint (6) limits formation flights during the daytime to an event number.

Constraint (7) limits formation flights during nighttime to an event number.

Constraint (8) ensures that a student is not paired with himself in formation flights.

Constraint (9) ensures that an instructor is not paired with himself in formation flights.

Constraint (10) limits the number of flight hours of the squadron to the "goal" hours of the day.

APPENDIX B. USMC TRAINEE MODEL DATA SETS AND OUTPUT ROUTINES

GAMS program listing of the Trainee model from Ref. 3

```
$TITLE MODEL 3 (TRAINEES) - USMC
$OFFUPPER OFFSYMXREF OFFSYMLIST
OPTIONS SOLPRINT = OFF
OPTIONS LIMCOL = 0, LIMROW = 0

*  An integer programming model for flight training scheduling in the USMC.
*  Daily flight schedule for trainees in combat capable training course will
*  be solved.
*  The items for tomorrow will be selected from the allowable set of items,
*  and qualified instructors will be paired, subject to both instructor and
*  flight hours ( aircraft ) availability.
*  MOE of the model is to keep the students on schedule.
*  Part of the data sets are obtained FRS HMT-303, USMC, Camp Pendleton, CA.

* Data sets
SETS
  I  items of syllabus (34)
                    / FAM100, FAM101, FAM102, FAM103, FAM104, FAM105
                      FAM106, FAM107, FAM108, FAM109, FAM110, FAM111
                      INS120, INS121, INS122, INS123, INS124, INS125
                      FOM130, FOM131, FOM132, TEF140, NAV150, NAV151
                      NAV152, ATG160, ATG161, ATG162, TAC170, TAC171
                      NVG180, NVG181, NVG182, CCX190                  /

  P  student pilots (11)
                    / DARLING, SHEERIN, STEININGER, PANTEN, HENSEL, MILNE
                      ADAMS,   ROSENTL, EAGLE,      READ,   KANG           /

  Q  instructors (15)
                    / GULMAN, CARPENTER,  WEIGL, CASTEEL, HALL,      KOLB
                      WEST,   SCHLESINGR, FORD,  JONES,   HENDRICK,  OWENS
                      GRACE,  EMERY,      ORNER                               / ;

PARAMETERS
  HBAR(Q)  maximum flight hours per day for instructor Q
          / GULMAN    3, CARPENTER  0,  WEIGL     0, CASTEEL      2
            HALL      4, KOLB       0,  WEST    1.5, SCHLESINGR   3
            FORD      4, JONES      3,  HENDRICK 2,  OWENS        3
            GRACE     4,  EMERY     1,  ORNER    2                  /

  H(I)  training time allowed for each syllabus flight I
     / FAM100 1.5,  FAM101 1.5,  FAM102 1.5,  FAM103 2.0,  FAM104 2.0
       FAM105 2.0,  FAM106 2.0,  FAM107 2.0,  FAM108 2.0,  FAM109 1.5
       FAM110 1.5,  FAM111 2.0,  INS120 1.5,  INS121 1.5,  INS122 1.5
       INS123 2.0,  INS124 2.0,  INS125 1.5,  FOM130 1.0,  FOM131 1.0
       FOM132 1.0,  TEF140 1.5,  NAV150 1.5,  NAV151 1.5,  NAV152 1.5
       ATG160 1.5,  ATG161 1.5,  ATG162 1.5,  TAC170 1.5,  TAC171 1.5
       NVG180 1.5,  NVG181 1.5,  NVG182 1.5,  CCX190 2.0                /

  IBAR(P) maximum number of training items per day for pilot P
     / DARLING  2, SHEERIN  2, STEININGER  2, PANTEN  2, HENSEL  2
       MILNE    2, ADAMS    2, ROSENTL     2, EAGLE   2, READ    2
       KANG     2    /
```

```
DHAT(P) number of days that pilot P has been assigned for training
    / DARLING   42, SHEERIN   42, STEININGER  42, PANTEN   70
      HENSEL    42, MILNE     70, ADAMS       70, ROSENTL   7
      EAGLE      7, READ       7, KANG         7               /

NC(P) number of finished items since assignment of pilot P
    / DARLING 11,   SHEERIN 12,   STEININGER 17,  PANTEN 23
      HENSEL  11,   MILNE   21,   ADAMS       8,  ROSENTL 1
      EAGLE    2,   READ     0,   KANG        1               / ;

TABLE
  QUAL(I,Q)  qualification of instructor to teach item I
        GULMAN CARPENTER WEIGL CASTEEL  HALL  KOLB   WEST SCHLESINGR
FAM100    1       1        1      1      1     1      1      1
FAM101    1       1        1      1      1     1      1      1
FAM102    1       1        1      1      1     1      1      1
FAM103    1       1        1      1      1     1      1      1
FAM104    1       1        1      1      1     1      1      1
FAM105    1       1        1      1      1     1      1      1
FAM106    1       1        1      1      1     1      1      1
FAM107    1       1        1      1      1     1      1      1
FAM108    1       1        1      1      1     1      1      1
FAM109    1       1        1      1      1     1      1      1
FAM110    1       1        1      1      1     1      1      1
FAM111    1       1        1      1      1     1      1      1
INS120    1       1        1      1      1     1      1      1
INS121    1       1        1      1      1     1      1      1
INS122    1       1        1      1      1     1      1      1
INS123    1       1        1      1      1     1      1      1
INS124    1       1        1      1      1     1      1      1
INS125    1       1        1      1      1     1      1      1
FOM130    1       1        1      0      1     1      1      1
FOM131    1       1        1      0      1     1      1      1
FOM132    1       1        1      0      1     1      1      1
TEF140    1       1        1      1      1     1      1      1
NAV150    1       1        1      1      1     1      1      1
NAV151    1       1        1      1      1     1      1      1
NAV152    1       1        1      1      1     1      1      1
ATG160    1       1        1      1      1     1      1      1
ATG161    1       1        1      1      1     1      1      1
ATG162    1       1        1      1      1     1      1      1
TAC170    1       1        1      1      1     1      1      1
TAC171    1       1        1      1      1     1      1      1
NVG180    1       0        1      1      0     1      1      1
NVG181    1       0        1      1      0     1      1      1
NVG182    1       0        1      1      0     1      1      1
CCX190    1       0        1      1      0     1      1      1

  +      FORD    JONES   HENDRICK OWENS   GRACE   EMERY   ORNER
FAM100    1        1        1       1       1       0       1
FAM101    1        1        1       1       1       0       1
FAM102    1        1        1       1       1       0       1
FAM103    1        1        1       1       1       0       1
FAM104    1        1        1       1       1       0       1
FAM105    1        1        1       1       1       0       1
FAM106    1        1        1       1       1       0       1
FAM107    1        1        1       1       1       0       1
FAM108    1        1        1       1       1       0       1
FAM109    1        1        1       1       1       0       1
FAM110    1        1        1       1       1       0       1
FAM111    1        1        1       1       1       0       1
```

```
INS120   1      1      1      1      1      1      1
INS121   1      1      1      1      1      1      1
INS122   1      1      1      1      1      1      1
INS123   1      1      1      1      1      1      1
INS124   1      1      1      1      1      1      1
INS125   1      1      1      1      1      1      1
FOM130   1      1      1      1      1      1      1
FOM131   1      1      1      1      1      1      1
FOM132   1      1      1      1      1      1      1
TEF140   1      1      1      1      1      0      1
NAV150   0      1      1      1      1      0      1
NAV151   0      1      1      1      1      0      1
NAV152   0      1      1      1      1      0      1
ATG160   1      1      1      1      1      0      1
ATG161   1      1      1      1      1      0      1
ATG162   1      1      1      1      1      0      1
TAC170   1      1      1      1      1      0      1
TAC171   1      1      1      1      1      0      1
NVG180   1      1      1      1      1      0      1
NVG181   1      1      1      1      1      0      1
NVG182   1      1      1      1      1      0      1
CCX190   1      1      1      1      1      0      1   ;


TABLE
  PROG(I,P)  completed items I for student pilot P
        DARLING SHEERIN STEININGER PANTEN HENSEL MILNE ADAMS
 FAM100   1       1         1        1      1      1     1
 FAM101   1       1         1        1      1      1     1
 FAM102   1       1         1        1      1      1     1
 FAM103   1       1         1        1      1      1     1
 FAM104   1       1         1        1      1      1     1
 FAM105   1       1         1        1      0      1     1
 FAM106   1       1         1        1      0      1     0
 FAM107   0       1         1        1      0      1     0
 FAM108   0       1         1        1      0      1     0
 FAM109   0       0         1        1      0      1     0
 FAM110   0       0         0        1      0      1     0
 FAM111   0       0         0        1      0      0     0
 INS120   1       1         1        1      1      1     0
 INS121   0       1         1        1      1      1     0
 INS122   1       0         1        0      1      1     0
 INS123   0       0         0        0      1      0     0
 INS124   0       0         0        1      0      0     0
 INS125   0       0         0        0      0      0     0
 FOM130   0       0         0        0      0      0     0
 FOM131   0       0         0        0      0      0     0
 FOM132   0       0         0        0      0      0     0
 TEF140   1       0         1        1      0      1     0
 NAV150   1       0         1        1      1      1     0
 NAV151   0       1         1        1      0      1     0
 NAV152   0       0         1        0      0      0     0
 ATG160   0       0         0        1      0      1     1
 ATG161   0       0         0        1      0      1     1
 ATG162   0       0         0        1      0      1     0
 TAC170   0       0         0        0      0      0     0
 TAC171   0       0         0        0      0      0     0
 NVG180   0       0         0        1      0      1     0
 NVG181   0       0         0        1      0      0     0
 NVG182   0       0         0        0      0      0     0
 CCX190   0       0         0        0      0      0     0
```

```
+         ROSENTL  EAGLE     READ      KANG
FAM100     1        1         0         1
FAM101     0        1         0         0
FAM102     0        0         0         0
FAM103     0        0         0         0
FAM104     0        0         0         0
FAM105     0        0         0         0
FAM106     0        0         0         0
FAM107     0        0         0         0
FAM108     0        0         0         0
FAM109     0        0         0         0
FAM110     0        0         0         0
FAM111     0        0         0         0
INS120     0        0         0         0
INS121     0        0         0         0
INS122     0        0         0         0
INS123     0        0         0         0
INS124     0        0         0         0
INS125     0        0         0         0
FOM130     0        0         0         0
FOM131     0        0         0         0
FOM132     0        0         0         0
TEF140     0        0         0         0
NAV150     0        0         0         0
NAV151     0        0         0         0
NAV152     0        0         0         0
ATG160     0        0         0         0
ATG161     0        0         0         0
ATG162     0        0         0         0
TAC170     0        0         0         0
TAC171     0        0         0         0
NVG180     0        0         0         0
NVG181     0        0         0         0
NVG182     0        0         0         0
CCX190     0        0         0         0          ;
```

```
ALIAS(I,J);

TABLE
 PREREQ(I,J) item I is prerequisite for item J
          FAM100  FAM101  FAM102  FAM103  FAM104  FAM105  FAM106  FAM107
 FAM100    0       1       1       1       1       1       1       1
 FAM101    0       0       1       1       1       1       1       1
 FAM102    0       0       0       1       1       1       1       1
 FAM103    0       0       0       0       1       1       1       1
 FAM104    0       0       0       0       0       1       1       1
 FAM105    0       0       0       0       0       0       1       1
 FAM106    0       0       0       0       0       0       0       1
 FAM107    0       0       0       0       0       0       0       0
 FAM108    0       0       0       0       0       0       0       0
 FAM109    0       0       0       0       0       0       0       0
 FAM110    0       0       0       0       0       0       0       0
 FAM111    0       0       0       0       0       0       0       0
 INS120    0       0       0       0       0       0       0       0
 INS121    0       0       0       0       0       0       0       0
 INS122    0       0       0       0       0       0       0       0
 INS123    0       0       0       0       0       0       0       0
 INS124    0       0       0       0       0       0       0       0
 INS125    0       0       0       0       0       0       0       0
 FOM130    0       0       0       0       0       0       0       0
```

FOM131	0	0	0	0	0	0	0	0
FOM132	0	0	0	0	0	0	0	0
TEF140	0	0	0	0	0	0	0	0
NAV150	0	0	0	0	0	0	0	0
NAV151	0	0	0	0	0	0	0	0
NAV152	0	0	0	0	0	0	0	0
ATG160	0	0	0	0	0	0	0	0
ATG161	0	0	0	0	0	0	0	0
ATG162	0	0	0	0	0	0	0	0
TAC170	0	0	0	0	0	0	0	0
TAC171	0	0	0	0	0	0	0	0
NVG180	0	0	0	0	0	0	0	0
NVG181	0	0	0	0	0	0	0	0
NVG182	0	0	0	0	0	0	0	0
CCX190	0	0	0	0	0	0	0	0

+	FAM108	FAM109	FAM110	FAM111	INS120	INS121	INS122	INS123
FAM100	1	1	1	1	1	1	1	1
FAM101	1	1	1	1	1	1	1	1
FAM102	1	1	1	1	1	1	1	1
FAM103	1	1	1	1	1	1	1	1
FAM104	1	1	1	1	1	1	1	1
FAM105	1	1	1	1	0	0	0	0
FAM106	1	1	1	1	0	0	0	0
FAM107	1	1	1	1	0	0	0	0
FAM108	0	1	1	1	0	0	0	0
FAM109	0	0	1	1	0	0	0	0
FAM110	0	0	0	1	0	0	0	0
FAM111	0	0	0	0	0	0	0	0
INS120	0	0	0	0	0	1	0	1
INS121	0	0	0	0	0	0	0	0
INS122	0	0	0	0	0	0	0	0
INS123	0	0	0	0	0	0	0	0
INS124	0	0	0	0	0	0	0	0
INS125	0	0	0	0	0	0	0	0
FOM130	0	0	0	0	0	0	0	0
FOM131	0	0	0	0	0	0	0	0
FOM132	0	0	0	0	0	0	0	0
TEF140	0	0	0	0	0	0	0	0
NAV150	0	0	0	0	0	0	0	0
NAV151	0	0	0	0	0	0	0	0
NAV152	0	0	0	0	0	0	0	0
ATG160	0	0	0	0	0	0	0	0
ATG161	0	0	0	0	0	0	0	0
ATG162	0	0	0	0	0	0	0	0
TAC170	0	0	0	0	0	0	0	0
TAC171	0	0	0	0	0	0	0	0
NVG180	0	0	0	0	0	0	0	0
NVG181	0	0	0	0	0	0	0	0
NVG182	0	0	0	0	0	0	0	0
CCX190	0	0	0	0	0	0	0	0

+	INS124	INS125	FOM130	FOM131	FOM132	TEF140	NAV150	NAV151
FAM100	1	1	1	1	1	1	1	1
FAM101	1	1	1	1	1	1	1	1
FAM102	1	1	1	1	1	1	1	1
FAM103	1	1	1	1	1	1	1	1
FAM104	1	1	1	1	1	1	1	1
FAM105	0	0	0	0	0	0	0	0
FAM106	0	0	0	0	0	0	0	0

FAM107	0	0	0	0	0	0	0	0
FAM108	0	0	0	0	0	0	0	0
FAM109	0	0	0	0	0	0	0	0
FAM110	0	0	0	0	0	0	0	0
FAM111	0	0	0	0	0	0	0	0
INS120	1	1	0	0	0	1	0	0
INS121	0	1	0	0	0	0	0	0
INS122	0	1	0	0	0	0	0	0
INS123	0	1	0	0	0	0	0	0
INS124	0	1	0	0	0	0	0	0
INS125	0	0	0	0	0	0	0	0
FOM130	0	0	0	1	1	1	0	0
FOM131	0	0	0	0	1	1	0	0
FOM132	0	0	0	0	0	1	0	0
TEF140	0	0	0	0	0	0	0	0
NAV150	0	0	0	0	0	0	0	0
NAV151	0	0	0	0	0	0	0	0
NAV152	0	0	0	0	0	0	0	0
ATG160	0	0	0	0	0	0	0	0
ATG161	0	0	0	0	0	0	0	0
ATG162	0	0	0	0	0	0	0	0
TAC170	0	0	0	0	0	0	0	0
TAC171	0	0	0	0	0	0	0	0
NVG180	0	0	0	0	0	0	0	0
NVG181	0	0	0	0	0	0	0	0
NVG182	0	0	0	0	0	0	0	0
CCX190	0	0	0	0	0	0	0	0

+	NAV152	ATG160	ATG161	ATG162	TAC170	TAC171	NVG180	NVG181
FAM100	1	1	1	1	1	1	1	1
FAM101	1	1	1	1	1	1	1	1
FAM102	1	1	1	1	1	1	1	1
FAM103	1	1	1	1	1	1	1	1
FAM104	1	1	1	1	1	1	1	1
FAM105	0	0	0	0	0	0	0	0
FAM106	0	1	1	1	1	1	0	0
FAM107	0	0	0	0	0	0	0	0
FAM108	0	0	0	0	0	0	0	0
FAM109	0	0	0	0	0	0	1	1
FAM110	0	0	0	0	0	0	1	1
FAM111	0	0	0	0	0	0	0	0
INS120	0	0	0	0	0	0	0	0
INS121	0	0	0	0	0	0	0	0
INS122	0	0	0	0	0	0	0	0
INS123	0	0	0	0	0	0	0	0
INS124	0	0	0	0	0	0	0	0
INS125	0	0	0	0	0	0	0	0
FOM130	0	0	0	0	0	0	0	0
FOM131	0	0	0	0	0	0	0	0
FOM132	0	0	0	0	0	0	0	0
TEF140	0	1	1	1	0	0	0	0
NAV150	1	0	0	0	0	0	1	1
NAV151	1	0	0	0	0	0	0	0
NAV152	0	0	0	0	0	0	0	0
ATG160	0	0	0	1	1	1	0	0
ATG161	0	0	0	0	1	1	0	0
ATG162	0	0	0	0	1	1	0	0
TAC170	0	0	0	0	0	1	0	0
TAC171	0	0	0	0	0	0	0	0
NVG180	0	0	0	0	0	0	0	0
NVG181	0	0	0	0	0	0	0	0

```
NVG182   0      0      0      0      0      0      0      0
CCX190   0      0      0      0      0      0      0      0


+       NVG182  CCX190
FAM100   1       1
FAM101   1       1
FAM102   1       1
FAM103   1       1
FAM104   1       1
FAM105   0       1
FAM106   0       1
FAM107   0       1
FAM108   0       1
FAM109   1       1
FAM110   0       1
FAM111   0       1
INS120   0       1
INS121   0       1
INS122   0       1
INS123   0       1
INS124   0       1
INS125   0       1
FOM130   0       1
FOM131   0       1
FOM132   0       1
TEF140   0       1
NAV150   1       1
NAV151   0       1
NAV152   0       1
ATG160   0       1
ATG161   0       1
ATG162   0       1
TAC170   0       1
TAC171   0       1
NVG180   0       1
NVG181   0       1
NVG182   0       1
CCX190   0       0     ;

SCALAR
  NTOTAL   total number of syllabus items in the course       /  34 /
  DTOTAL   total number of days pilot is allowed for training / 140 /
  CPRIME   weight for penalty variable Z                      /   1 /
  FTHR     flight hour goal on a day T                        /  24 / ;

* Mathematical model removed. Full GAMS USMC model provided in Ref 3.


* Output procedures
PARAMETER
  TRAINING(I,P)   Items and Students
  SECOND(I,P)     exclusive second item
  TEACHER(I,P,Q)  Items - Students and Instructors ;
    TRAINING(I,P)  = X.L(I,P) $ IP(I,P)     ;
    SECOND(I,P)    = X.L(I,P) $ IE(I,P)     ;
    TEACHER(I,P,Q) = Y.L(I,P,Q) $ IQ(I,P,Q) ;

*print out the solution in a tabular format
DISPLAY TRAINING ;
DISPLAY SECOND ;
DISPLAY TEACHER ;
```

```
DISPLAY ZP.L ;
DISPLAY ZM.L ;
DISPLAY W.L ;
```

APPENDIX C. GAMS DATA ELEMENTS FOR THE USMC TRAINEE MODEL

I
- Non-Key Data
- EventName

P
- Non-Key Data
- StudentName

Q
- Non-Key Data
- InstructorName

H(I)
- Non-Key Data
- EventName
- TrainingTime

IBAR(P)
- Non-Key Data
- StudentName
- TrainingItemsPerDay

HBAR(Q)
- Non-Key Data
- InstructorName
- MaxFltHrsPerDay

DHAT(P)
- Non-Key Data
- StudentName
- DaysAssigned

NC(P)
- Non-Key Data
- StudentName
- FinishedItems

QUAL(I,Q)
- Non-Key Data
- EventName
- InstructorName
- Qualficiation

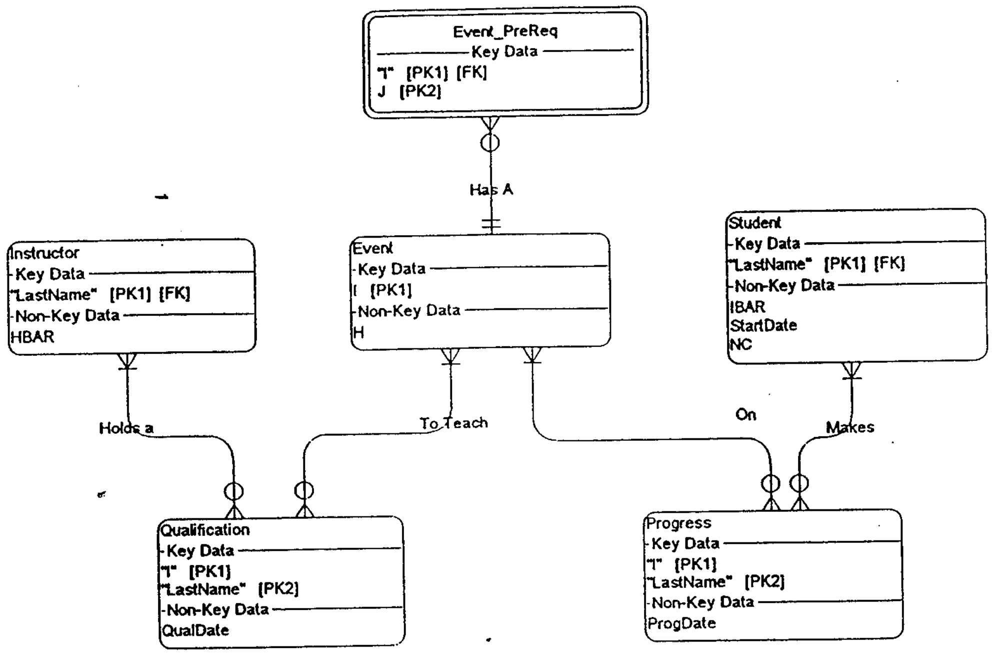
PROG(I,P)
- Non-Key Data
- EventName
- StudentName
- Progress

PREREQ(I,J)
- Non-Key Data
- EventName
- EventName
- Prerequisite

Legend:

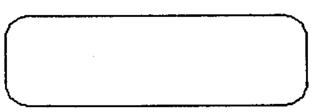
<u>Symbol</u>	<u>Meaning</u>
(rounded box)	An entity

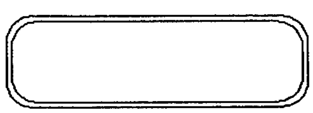
APPENDIX D. USMC TRAINEE MODEL IN THIRD NORMAL FORM



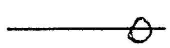
Legend:

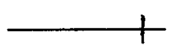
Symbol | Meaning

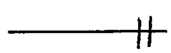

An entity


A weak entity


A relationship with a cardinality of many


An optional relationship


A mandatory relationship


A one and only one relationship

APPENDIX E. USMC TRAINEE GAMS MODEL REQUIRED INPUTS

A. Sets

GAMS Input	Database File	Data Input	Query Name
I	EVENT.DB	I	I.SQL
P	STUDENT.DB	P	P.SQL
Q	INSTRUCTOR.DB	Q	Q.SQL

B. Parameters

GAMS Input	Database File	Data Input	Query Name
HBAR(Q)	INSTRUCTOR.DB	Q, HBAR	HBAR.SQL
H(I)	EVENT.DB	I, H	H.SQL
IBAR(P)	STUDENT.DB	P, IBAR	IBAR.SQL
DHAT(P)	STUDENT.DB	P, DHAT	DHAT.SQL
NC(P)	STUDENT.DB	P, NC	NC.SQL

C. Tables

GAMS Input	Database File	Data Input	Query Name
QUAL(I,Q)	QUALIFICATION.DB	I, Q, QUAL	QUALT.SQL
PROG(I,P)	PROG.DB	I, P, PROG	PROGT.SQL
PREREQ(I,J)	EVENT_PRE.DB	I, J, PREREQ	PREREQT.SQL

APPENDIX F. REQUIRED QUERIES

DHAT.SQL

```
SELECT DISTINCT P, DHAT
FROM "STUDENT.DB"
```

H.SQL

```
SELECT DISTINCT I, H
FROM "EVENT.DB"
```

HBAR.SQL

```
SELECT DISTINCT Q, HBAR
FROM "INSTRUCT.DB"
```

I.SQL

```
SELECT DISTINCT I
FROM "EVENT.DB"
```

IBAR.SQL

```
SELECT DISTINCT P, IBAR
FROM "STUDENT.DB"
```

NC.SQL

```
SELECT DISTINCT P, DHAT
FROM "PROG.DB"
```

P.SQL

```
SELECT DISTINCT P
FROM "STUDENT.DB"
```

PREQT.SQL

```
SELECT DISTINCT I, J, PREREQ
FROM "EVE_PRE.DB"
```

PROGT.SQL

```
SELECT DISTINCT I, P, PROG
FROM "PROG.DB"
```

Q.SQL

```
SELECT DISTINCT Q
FROM "INSTRUCT.DB"
```

QUALT.SQL

```
SELECT DISTINCT I, Q, QUAL
FROM "QUAL.DB"
```

APPENDIX G. MODIFIED USMC GAMS TRAINEE MODEL DATA SETS AND OUTPUT ROUTINES

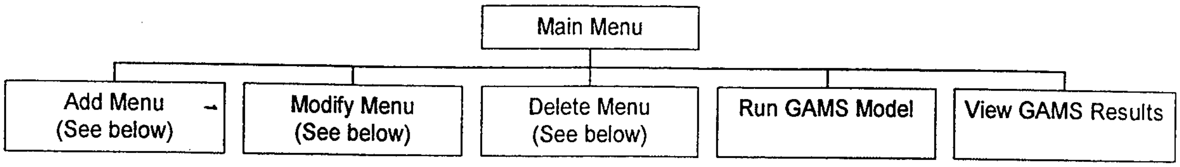
```
$TITLE MODEL 3 (TRAINEES) - USMC
$OFFUPPER OFFSYMXREF OFFSYMLIST
OPTIONS SOLPRINT = OFF
OPTIONS LIMCOL = 0, LIMROW = 0

* Data sets
SETS
  I  items of syllabus
/
$include "c:\flight\I.txt"
/

  P  student pilots
/
$include "c:\flight\P.txt"
/

  Q  instructors
/
$include "c:\flight\Q.txt"
/ ;

PARAMETERS
  HBAR(Q) maximum flight hours per day for instructor Q
/
$include "c:\flight\hbar.txt"
/

  H(I)  training time that is needed for item I
/
$include "c:\flight\h.txt"
/

  IBAR(P) maxnumber of training items per day for pilot P
/
$include "c:\flight\ibar.txt"
/
```

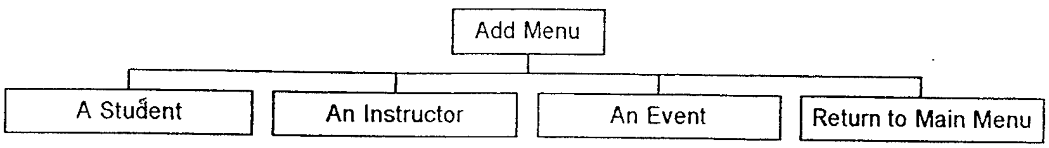
```
  DHAT(P) number of days pilot P is assigned for training
/
$include "c:\flight\dhat.txt"
/

  NC(P) number of finished items for pilot P
/
$include "c:\flight\nc.txt"
/ ;

PARAMETER
  QUAL(I,Q)  qualification of instructor to teach item I
/
$include "c:\flight\qualt.txt"
/  ;
PARAMETER
  PROG(I,P) completed items I for student pilot P
/
$include "c:\flight\progt.txt"
/  ;

ALIAS(I,J);

PARAMETER
  PREREQ(I,J) item I is prerequisite for item J
/
$include "c:\flight\prereqt.txt"
/  ;

SCALAR
NTOTAL   total number of syllabus items in the course
/
$include "c:\flight\ntotal.txt"
/;

SCALAR
DTOTAL   total number of days pilot is allowed for training
/
$include "c:\flight\dtotal.txt"
/;
```

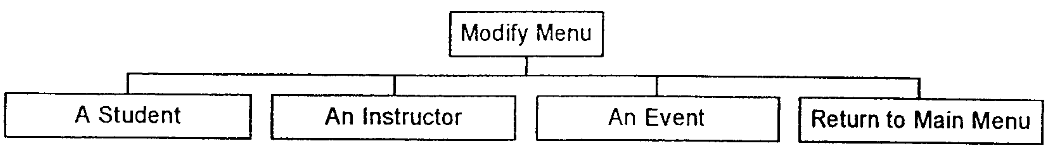
```
SCALAR
CPRIME   weight for penalty variable Z
/
$include "c:\flight\cprime.txt"
/;

SCALAR
FTHR     flight hour goal on a day T
/
$include "c:\flight\fthr.txt"
/;

* Mathematical model removed.  Full GAMS USMC model provided in Ref 3.

* Output proceudres
PARAMETER
  TRAINING(I,P)   Items and Students
  SECOND(I,P)     exclusive second item
  TEACHER(I,P,Q)  Items - Students and Instructors ;
  TRAINING(I,P)   = X.L(I,P) $ IP(I,P)     ;
  SECOND(I,P)     = X.L(I,P) $ IE(I,P)     ;
  TEACHER(I,P,Q) = Y.L(I,P,Q) $ IQ(I,P,Q) ;

* print out the solution in a tabular format
* DISPLAY TRAINING ;
FILE TRN /'c:\flight\TRAINING.TXT'/;
PUT TRN;
TRN.PC=5;
LOOP( IP(I,P)$(X.L(I,P) ), PUT P.TL, I.TL /; ) ;

* DISPLAY SECOND ;
FILE SEC /'c:\flight\second.TXT'/;
PUT SEC;
SEC.PC=5;
LOOP( IE(I,P)$(X.L(I,P) ), PUT P.TL, I.TL /; ) ;
```

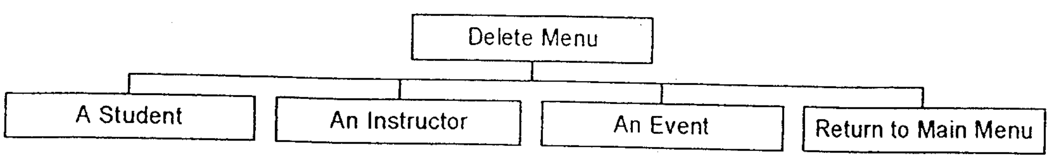
```
* DISPLAY TEACHER
FILE TEA /'c:\flight\TEACHER.TXT'/;
PUT TEA;
TEA.PC=5;
LOOP( IQ(I,P,Q)$(Y.L(I,P,Q) ), PUT Q.TL, P.TL, I.TL /; ) ;

* DISPLAY ZP.L ;
* DISPLAY ZM.L ;
* DISPLAY W.L ;
```

APPENDIX H. HIERARCHICAL MENU STRUCTURE









INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5101	2
3.	Hemant Bhargava (Code SM/BH) Naval Postgraduate School Department of Systems Management Monterey, California 93943-5103	2
4.	R. Kevin Wood (Code OR/WD) Naval Postgraduate School Department of Operations Research Monterey, California 93940-5008	4
5.	Richard Rosenthal (Code OR/RL) Naval Postgraduate School Department of Operations Research Monterey, California 93940-5008	1
6.	U.S. Army AI Center DIRECTOR INFO SYS CMD CON COMM COMP 107 Army Pentagon Washington, DC 20310-0107	1
7.	MR. Clark Pritchett U.S. Coast Guard R&D Center 1082 Shennecossette Ave. Groton, Connecticut 06340	1
8.	A.M. Geoffrion Graduate School of Management University of California 405 Hilgard Ave. Los Angeles, California 90024-1418	1

9. LT Michael Downs 2
234 Dundee
Monterey, California 93940